IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re: Application of: Mark Allen Grubbs	:
	: Before the Examiner:
Serial No: 10/621,951	: Charles Edward Lu
Filed: 07/17/2003	: Group Art Unit: 2163
Title: PERFORMANCE-ENHANCING SYSTEM AND METHOD OF ACCESSING FILE SYSTEM OBJECTS	: Confirmation No.: 1546

APPELLANTS' BRIEF UNDER 37 C.F.R. 1.192

Assistant Commissioner of Patents
Washington, D. C. 20231

Sir:

This is an appeal to a final rejection dated July 21, 2006 of Claims 1 - 20 in the Application. This brief is submitted pursuant to a Notice of Appeal filed on October 23, 2006 in accordance with 37 C.F.R. 1.192.

BRIEF FOR APPLICANTS - APPELLANTS

(i)

Real Party in Interest

The real party in interest is International Business Machines Corporation (IBM), the assignee.

(ii)

Related Appeals and Interferences

There are no other appeals or interferences known to appellants, appellants' representative or assignee, which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(iii)

Status of Claims

Claims 1 - 20 are finally rejected and Claims 1 – 20 are being appealed.

(iv)

Status of Amendment

An "Amendment after Final" was not filed.

(v)

Summary of Claimed Subject Matter

The Application contains three sets of claims. The first set (i.e., Claims 1 – 7) is a set of method claims. The second set (i.e., Claims 8 – 14) is a set of computer program product claims. The third set (i.e., Claims 15 - 20) is a set of system claims. The independent claim in each set of claims (i.e., Claims 1, 8 and 15) are of equivalent scope.

The invention, as claimed in Claims 1, 8 and 15 provides a performance-enhancing way of accessing frequently-accessed file system objects (see page 1, lines 9 – 11, page 5, lines 3 and 4). Accordingly, when mounting a file system at a mount point on a computer system, it is determined if there is at least one frequently-accessed file system object in the file system (see page 11, lines 19 – 31, page 12, lines 9 – 24, page 15, line 24 to page 12, line 7). Note that a frequently-accessed file system object is a file object that has been accessed a threshold number of times within a time frame (see page 14, line 31 to page 15, line 13). As customary, each file system object has a pathname and an inode number (see page 1, lines 14 – 17, page 1, line 28 to page 2, line 4). The inode number is used to locate the file system object on a storage system. If there is a frequently-accessed file system object, the pathname of the file system object is entered into a memory system. The pathname of the file system is also cross-referenced with its inode number in the memory system to enable the inode number to be obtained with one memory access (see page 11, lines 19 – 31 and items 1405 and 1410 in Fig. 14).

Note that in independent Claim 8, the code means plus functions are the steps in Figs. 15 and on page 15, line 21 to page 16, line 7.

(vi)

Grounds of Rejection to be Reviewed on Appeal

Whether it was proper to reject Claims 1, 4, 6 – 8, 11, 13 – 15, 18 and 20 under 35 U.S.C. §103(a) as being unpatentable over Sinha in view of Pinkoski and further in view of Achiwa et al.

Whether it was proper to reject Claims 2, 3, 9, 10, 16 and 17 under 35 U.S.C. §103(a) as being unpatentable over Sinha in view of Pinkoski further in view of Achiwa and further in view of Nevarez and further in view of Bauer

Whether it was proper to reject Claims 5, 12 and 19 under 35 U.S.C. §103(a) as being unpatentable over Sinha in view of Pinkoski further in view of Achiwa and further in view of Falkner

(vii)

<u>Arguments</u>

It is a well settled law that in considering a Section §103 rejection, the subject matter of the claim "as a whole" must be considered and analyzed. In the analysis, it is necessary that the scope and contents of the prior art and differences between the art and the claimed invention be determined. *Graham v. John Deere Co.*, 383 U.S. 1 (1966).

Whether it was proper to reject Claims 1, 4, 6 – 8, 11, 13 – 15, 18 and 20 under 35 U.S.C. §103(a) as being unpatentable over Sinha in view of Pinkoski and further in view of Achiwa et al.

<u>Claims 1, 8 and 15</u>

Sinha purports to teach a pathname resolution method for providing fixed speed of file accessing in a distributed system. According to Sinha, various proposals have been made in the prior art for reducing the number of network accesses which are necessary to achieve path name resolution in a distributed system. One method is to use a name cache at each node, which relates path names to the locations of the corresponding files within the distributed system. By using such a name cache, it becomes unnecessary to execute multiple network accesses in order to achieve path name resolution.

However, the amount of main memory available at each node of the distributed system is limited, the size of the memory region available for a name cache is thus small. Hence, when a maximum number of entries of the name cache is exceeded, it becomes necessary to perform replacement processing at each node, thereby deleting one or more entries to make room for new entries. Therefore, it is impossible for a user to forecast the amount of time which will be required to access any specific file within the distributed system. Due to that fact, it is difficult to use a name cache in a real-time processing system.

Consequently, Sinha teaches a method whereby each user of a distributed system can selectively specify one of a plurality of modes of path name resolution for use in accessing a specific file from a node of the system. A selected mode provides a fixed, minimum access time for the file and utilizes a name cache within the main memory of that node. In one of the selected modes, the name cache contains pathnames of frequently accessed files which are cross-referenced to the location of the files. For example, if the file is on a remote system, the location of the file will contain an identification of the remote system and information to locate the file on that remote system.

But note that it is when the user decides that a file will be frequently accessed that the name of the file is entered into the name cache (see col. 7, lines 41 – 54). And, when the user decides the file will no longer be used frequently, the entry is deleted from the memory (see col. 8, lines 7 – 14). By contrast, the present invention enters the pathname of the file with its cross-referenced inode number in the memory system once it is determined that the file will be frequently accessed and at the time ***the file system is mounted at a mount point on the computer system***.

Further, the claimed invention includes the limitations of cross-referencing the pathname of the file to its inode number in the memory system. Sinha, on the other hand, teaches that the pathname of the file is to be cross-referenced to the node or computer system on which it is stored (see col. 8, lines 58 – 66 and Fig. 4b of Sinha).

The Examiner acknowledged that Sinha does not teach the step of cross-referencing the pathname of the file to its inode number but rather to its node identifier. Due to this discrepancy, the Examiner used Pinkowski to show "a conventional system where <u>nodes are inodes</u>". Consequently, the Examiner reasoned, since the nodes of Sinha and the inodes of Pinkowski both specify location information of a file, the teachings of Sinha can be combined with those of Pinkowski to teach the invention. Applicants disagree.

Firstly, the Examiner has mischaracterized Pinkowski. Pinkowski purports to teach a method and apparatus for file server addressing. According to the purported teachings of Pinkowski, a file handle in a call that accesses a file includes a file system identification, a file identification or inode number and a generation number. The addressing system converts the multi-bit file identification number into an alternative path name that identifies a location for the file without the need for converting the real path name. Specifically, the inode number in binary form is translated into a multiple digit hexadecimal form that is parsed into directory names. The last directory name provides the location of the designated file.

Thus, Pinkowski does not teach, show or suggest that ***nodes are inodes*** as claimed by the Examiner. Rather, Pinkowski teaches the use of a file identification or inode number to acces a file.

Secondly, Sinha specifically teaches storing the pathname of a file cross-referenced with the node identifier of the node on which it is stored in memory to facilitate quick accesses to the file. Therefore, the Examiner cannot replace the node identifier of Sinha with the inode number of Pinkowski in a quest to show that the combination of the references teaches the claimed invention. Put differently, the Examiner has impermissibly combined the teachings of Sinha with those of Pinkowski.

It should nevertheless be noted that an inode is not equivalent to a node. As is well known in the field as well as stated in the Specification, an inode is a data structure that contains important information about the file with which it is associated. Information contained in an inode includes user and group ownership of the file, access permissions (e.g., read, write, execute permissions) and file type (e.g., regular, directory or device file). Further, the inode contains the date and time the file was created as well as the date and time of any modifications. In addition, the inode contains information regarding the location of the file on a disk or storage system.

Therefore, an inode number and a node identifier are not interchangeable.

The Examiner further acknowledged that the teachings of Sinha and those of Pinkowski, alone or in combination, do not teach the step of entering the pathname of the file upon mounting the file system at a mount point on a computer system. However, the Examiner stated that “Achiwa et al. teaches processing upon mounting the file system at a mount point on a computer system” and therefore, it would have been obvious to combine the teachings of Achiwa et al. with those of Sinha and Pinkowski to arrive at the claimed invention.

Processing always occurs when a file system is mounted at a mount point. Thus, Applicants does not understand why “processing upon mounting a file system at a mount point” is a suggestion to combine the teachings of Achiwa et al. with those of Sinha and Pinkowski.

A showing of a suggestion, teaching, or motivation to combine the prior art references is an "essential evidentiary component of an obviousness holding." *C.R. Bard, Inc. v. M3 Sys. Inc.*, 157 F.3d 1340, 1352, 48 USPQ 2d 1225, 1232 (Fed. Cir. 1998). This evidence may flow from the prior art references themselves, the knowledge of one of ordinary skill in the art, or, in some cases, from the nature of the problem to be solved. See *Pro-Mold & Tool Co. v. Great Lakes Plastics, Inc.*, 75 F.3d 1568, 1573, 37 USPQ 2d 1626, 1630 (Fed. Cir. 1996). However, the suggestion more often comes from the teachings of the pertinent references. See *In re Rouffet*, 149 F.3d 1350, 1359, 47 USPQ 2d 1453, 1459 (Fed. Cir. 1998). This showing must be clear and particular, and broad conclusory statements about the teaching of multiple references, standing alone, are not "evidence." See *Dembiczak*, 175 F.3d at 1000, 50 USPQ2d at 1617.

Achiwa et al. purport to teach a method for file space management. According to Achiwa et al., the method provides for moving one or more files from a client storage system to a server storage system. Specifically, a file from the client storage system is copied to the server storage system and is deleted from the client storage system, thus recovering storage space in the client storage system. A logical reference is provided in the client storage system to allow access requests to be made to the file from the client storage system, even though the file has been deleted therefrom. The logical reference also allows for the file to be reproduced in the client storage system when needed.

Sinha, on the other hand, specifically teaches that the entry of the pathname of the file in the memory occurs when a user decides that the file will henceforth be used frequently (see col. 7, lines 41 – 54). And, when the user decides the file will no longer be used frequently, the entry is deleted from the memory (see col. 8, lines 7 – 14). Since, Sinha specifically teaches the time the pathname of a file is entered or deleted in the memory, there would not be any reason for combining the teachings of Achiwa et al. with those of the Sinha and Pinkowski as asserted by the Examiner to show that the pathname of a file is entered or deleted in the memory when the file system containing the file is mounted or unmounted, respectively.

Therefore, Applicants submit that Claims 1, 8 and 15 are patentable over the combination of the teachings of Sinha, Pinkowski and Achiwa et al.

<u>Claims 6, 13 and 20</u>

Claims 6, 13 and 20 contain the limitations that the pathname of the file system object and its cross-referenced inode number are removed from the memory system when the file system containing the file system object is unmounted.

As mentioned above, Sinha specifically teaches that when the user decides the file will no longer be used frequently, the entry is deleted from the memory (see col. 8, lines 7 – 14). Thus, according to the teachings of Sinha, the pathname of the file will be deleted from the memory whether or not the file system that contains the file is unmounted so long as the user decides that the file will no longer be used frequently.

Since the claims contain the limitations of deleting the pathname of the file when the file system containing the file is unmounted while Sinha teaches that the pathname of the file is deleted when the user says so, Applicants submit that Claims 6, 13 and 20 are patentable over the teachings of Sinha, Pinkowski and Achiwa et al.

Whether it was proper to reject Claims 2, 3, 9, 10, 16 and 17 under 35 U.S.C. §103(a) as being unpatentable over Sinha in view of Pinkoski further in view of Achiwa and further in view of Nevarez and further in view of Bauer

Claims 2, 9 and 16 recite the limitations that the determining step includes the step of obtaining from an extended attribute file a list of pathnames to be entered into the memory system, the extended attribute file being associated with the mounted file system.

Nevarez purports to teach a method for storing a database in extended attributes of a file system. Accordingly, a database containing management, maintenance, control, and other user-defined data that are associated with a directory entry of the file system is created. The database is stored inside the extended attribute or sets of extended attributes. Each extended attribute includes a header for defining the extended attribute that comprises a first plurality of fields. It also includes a plurality of records for storing data. In turn, each record of the plurality of records comprises a second plurality of fields. The extended attribute is identified by a string stored in a directory entry of a directory entry table for the file system. The string comprises a predetermined substring denoting the extended attribute having the database architecture of the present invention. The first plurality of fields of the extended attribute includes fields for storing a length of the header, a version number of the header, a revision number of the header, the number of extended attributes in the database, and the number of records in the plurality of records in the database. The second plurality of fields includes fields for storing a name of a record, data, a record length, a length of the record name, and a length of the data. Both the name and data are stored as strings.

However, Nevarez does not teach the step of ***obtaining from an extended attribute file a list of pathnames <u>to be entered into the memory system</u>, the extended attribute file being associated with the mounted file system***.

Bauer, on the other hand, purports to teach a method for operating a computer based file system. The computer-based file system permits a real-time user-selectable search request of previously stored data objects using file access system calls which use search criteria other than a file-name-substring matching. Search criteria include a criterion type and a criterion value which combination is also referred to as a search criterion type/value pair. The file access system calls include a purported file name containing one or more non-file-name-substring-based search criteria. Using the non-file-name-substring-based search technique, a user can do look-up-by-key, relational look-up, phonetic spelling look-up, and find-operator look-up to locate one or more stored data objects of any database or file system of the system.

However, just as in the case of Nevarez, Bauer does not teach the step of ***obtaining from an extended attribute file a list of pathnames <u>to be entered into the memory system</u>, the extended attribute file being associated with the mounted file system***.

Consequently, combining the teachings of Nevarez and Bauer with those of Sinha, Pinkowski and Achiwa et al. does not teach the invention as claimed in Claims 2, 9 and 16.

Nonetheless, it should be pointed out that Sinha specifically teaches that a user makes the determination as to whether a file will be frequently accessed. When the user makes the determination, the user supplies the pathname of the file to the system in order for the name of the file to be entered into the name cache (see col. 7, lines 41 – 54). By combining the teachings of Nevarez and Bauer with those of Sinha, Pinkowski and Achiwa et al. to show the invention as claimed in Claims 2, 9 and 16, the Examiner is asserting that the determining step includes the step of obtaining from an extended attribute file a list of pathnames to be entered into the memory system wherein the extended attribute file is associated with the mounted file system.

Applicants submit that if the user supplies the pathname of the file, there is no reason for the name of the file to be obtained from an extended attribute file. In other words, the teachings of Nevarez and Bauer cannot be combined with those of Sinha, Pinkowski and Achiwa et al. to show the claimed invention and the Examiner has impermissibly combined the teachings of the references.

Whether it was proper to reject Claims 5, 12 and 19 under 35 U.S.C. §103(a) as being unpatentable over Sinha in view of Pinkoski further in view of Achiwa and further in view of Falkner

Claims 5, 12 and 19 contain the limitations that the determining step includes the step of monitoring accesses to a file system object within a certain time span.

Falkner purports to teach a method of determining working sets by logging and simulating filesystem operations. Accordingly, the method includes activating a filesystem logging mechanism to monitor filesystem transactions performed on the computer by the user. Next, a file cache is provided under computer control for storing at least a portion of at least one computer file. Also, a filesystem log file is provided for storing records of filesystem transactions invoked by the computer. A user work cycle is then performed during which the filesystem logging mechanism monitors filesystem transactions invoked by the computer and stores records of certain filesystem transactions to the filesystem log file. Finally, the size of the file cache required to store the information cached during the work cycle is determined by processing the log file.

Thus according to Falkner, it is the file system that is being monitored for transactions thereto. By contrast, it is accesses to the file that is being monitored in the claimed invention.

Further, in Falkner the file system transactions are monitored in order to determine the size of the cache needed to store the information cached during the work cycle. By contrast the step of determining in the present invention is to figure out whether the file is a frequently-accessed file.

Consequently, there would not be any reason to combine the teachings of Falkner with those of the other references to arrive at the claimed invention.

However, even if the teachings of the references were to be combined together, the resulting combination would not teach the claimed invention because, as stated above, the combination of the teachings of Sinha, Pinkowski and Achiwa et al. does not teach the claims on which these claims are dependent.

Based on the foregoings, Applicants request passage to issue of all the claims in the Application.

Respectfully Submitted

By: [illegible]
Volel Emile
Attorney for Applicants
Registration No. 39,969
(512) 306-7969

(viii)

Claims Appendix

1. (Previously presented) A method of providing a performance-enhancing way of accessing frequently-accessed file system objects comprising the steps of:

   determining at least one frequently-accessed file system object in a file system upon mounting the file system at a mount point on a computer system, each file system object having a pathname and an inode number, the inode number for locating the file system object on a storage system;

   entering the pathname of the at least one file system object into a memory system; and

   cross-referencing the pathname of the at least one file system object in the memory system with its inode number thereby enabling the inode number to be obtained with one memory access.

2. (Previously presented) The method of Claim 3 wherein the pathnames in the extended attribute file are relative to the mount point.

3. (Previously presented) The method of Claim 1 wherein the determining step includes the step of obtaining from an extended attribute file a list of pathnames to be entered into the memory system, the extended attribute file being associated with the mounted file system.

4. (Previously presented) The method of Claim 1 wherein the determining step includes the step of obtaining the pathname from a user.

5. (Previously presented) The method of Claim 1 wherein the determining step includes the step of monitoring accesses to a file system object within a certain time span.

6. (Previously presented) The method of Claim 1 wherein the pathname of the file system object and its cross-referenced inode number are removed from the memory system when the file system containing the file system object is unmounted.

7. (Previously presented) The performance-enhancing method of Claim 1 wherein a pathname of a file system object and its cross-referenced inode number is removed from the memory system when a user so ordered.

8. (Previously presented) A computer program on a computer readable medium for enhancing performance of a system when frequently-accessed file system objects are being accessed comprising:

code means for determining at least one frequently-accessed file system object in a file system upon the file system being mounted at a mount point on the system, each file system object having a pathname and an inode number, the inode number for locating the file system object on a storage system;

code means for entering the pathname of the at least one file system object into a memory system; and

code means for cross-referencing the pathname of the at least one file system object in the memory system with its i-node number thereby allowing the inode number to be obtained with one memory access.

9. (Previously presented) The computer program of Claim 10 wherein the pathnames in the extended attribute file are relative to the mount point.

10. (Original) The computer program of Claim 8 wherein the determining code means includes code means for obtaining from an extended attribute file a list of pathnames to be entered into the memory system, the extended attribute file being associated with the mounted file system.

11. (Previously presented) The computer program of Claim 8 wherein the determining code means includes code means for obtaining obtaining the pathname from a user.

12. (Original) The computer program of Claim 8 wherein the determining code means includes code means for monitoring accesses to a file system object within a certain time span.

13. (Previously presented) The computer program of Claim 8 wherein the pathname of the file system object and its cross-referenced inode number are removed from the memory system when the file system containing the file system object is unmounted.

14. (Original) The computer program of Claim 8 wherein a pathname of a file system object and its cross-referenced inode number is removed from the memory system when a user so ordered.

15. (Previously presented) A system comprising:

at least one storage system for storing code data; and at least one processor for processing the code data to determine at least one frequently-accessed file system object in a file system upon the file system being mounted at a mount point on the system, each file system object having a pathname and an inode number, the inode number for locating the file system object on a storage system, to enter the pathname of the at least one file system object into a memory system, and to cross-reference the pathname of the at least one file system object in the memory system with its inode number thereby allowing the inode number to be obtained with one memory access.

16. (Previously presented) The system of Claim 17 wherein the pathnames in the extended attribute file are relative to the mount point.

17. (Original) The system of Claim 15 wherein the code data is further processed to obtain from an extended attribute file a list of pathnames to be entered into the memory system, the extended attribute file being associated with the mounted file system.

18. (Previously presented) The system of Claim 15 wherein the code data is further processed to obtain the pathname from a user.

19. (Original) The system of Claim 15 wherein the code data is further processed to monitor accesses to a file system object within a certain time span.

20. (Previously presented) The system of Claim 15 wherein the pathname of the file system object and its cross-referenced inode number are removed from the memory system when the file system containing the file system object is unmounted.

(ix)

Evidence Appendix

No evidence was submitted pursuant to 37 C.F.R. §§ 1.130, 1.131 and 1.132 nor was there any evidence entered by the Examiner relied upon by Appellants in this appeal.

(x)

Related Proceedings Appendix

There are no decisions rendered by a court or the Board that would have a bearing on the Board's decision in the pending appeal.